Design Wave 設計コンテスト2007

64点高速フーリエ変換回路 設計仕様書

和田知久

今回の設計課題は，ディジタル信号処理ではかならず登場する高速フーリエ変換（FFT：fast Fourier transform）回路の設計です．高速フーリエ変換は，離散フーリエ変換（DFT：discrete Fourier transform）を高速に計算する手法です．今回はIEEE 802.11a/g/nなどのワイヤレスLANでよく使われている64点のFFTとします．（筆者）

コンテストの題材として64点はちょうどよいサイズと考えています．本稿では，FFTについての詳細やディジタル回路による実現方法を説明します．これまで，FFTの式は知っていても物理的イメージがわかなかった方にとっても良い教材になると思います．

コンテストでは，HDL（VHDLもしくはVerilog HDL）による設計と論理合成を行います．FPGA向けに無償で提供されている設計ツールでも参加できます．HDL設計に興味のある方はどしどし参加してください．また，余裕のある方はFPGAなどへ実装すれば，その努力が認められ，高い評価が得られると思います．FPGAへの実装もぜひトライしてみてください．

1．FFT回路の構成

図1に今回想定するシステムのブロック図を示します．システムは大きく分けて，送信信号生成部（Sender）とSenderの出力信号を高速フーリエ変換するFFT部から構成されます．詳しくは後述しますが，複素数の信号を扱います．実数部分をIポート，虚数部分をQポートとします．64点の複素数データは，IポートとQポートを使ってそれぞれシリアルに入力されるものとします．FFTの結果もシリアルに出力されます．

図1には示していませんが，64点の先頭の信号を示すフラグ信号があります．この信号によって64点の先頭位置をFFT回路は把握することができます．

● 離散フーリエ変換と逆変換

離散フーリエ変換（DFT）は，スペクトル解析などでよく使用されます．

離散フーリエ変換のイメージを**図2**に示します．図の左側に示す$x(n)$は解析対象の信号で，1周期をN点の区間に分けて示したものです．これを離散フーリエ変換した結果が図の右側で，$k=1$に大きな成分が現れます．このことは，元信号$x(n)$に$k=1$の周波数成分が多く含まれることを意味します．このような解析をスペクトラム解析と呼びます．

図2をよく見ると，$k=N-1$にも大きな成分があること

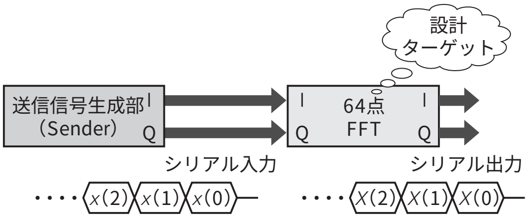


図1　システム・ブロック図
送信信号生成部（Sender）とSenderの出力信号を高速フーリエ変換するFFT部から構成される．

KeyWord　高速フーリエ変換，FFT，離散フーリエ変換，DFT，逆変換，回転因子

がわかります．DFTでは，N点の点が周期的につながっている関係にあり，$k = N - 1$は実は$k = -1$の周波数を示すことになります．負の周波数は理解しにくいかもしれませんが，$x(n)$の1周期の波形は，$k = -1$の周波数と$k = 1$の周波数成分からできていると考えることができます．

離散フーリエ変換（DFT）とその逆変換である逆離散フーリエ変換（IDFT）の式を以下に示します．

●DFT（FFT）

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0,\ 1,\ \ldots,\ N-1)$$

●IDFT（IFFT）

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) \cdot e^{j\left(\frac{2\pi}{N}\right)nk} \quad (n = 0,\ 1,\ \ldots,\ N-1)$$

…………………………(1)

N点の$x(n)$信号をN点の$X(k)$信号に変換するのがDFTであり，その逆がIDFTです．DFTを高速に演算する方式が高速フーリエ変換（FFT）であり，IDFTを高速に演算する方法が逆高速フーリエ変換（IFFT）です．以下では，すべてFFT，IFFTと呼ぶことにします．ただし，FFTでは，Nは2のべき乗という制約があります．

FFTとIFFTの式をよく見ると，二つの違いがあります．

1）最初に$1/N$があるかないか

2）指数のjの前の符号

したがって，二つの変換は非常によく似たものであり，**図3**に示すように，FFT回路があれば，IFFTを簡単に実現できます．

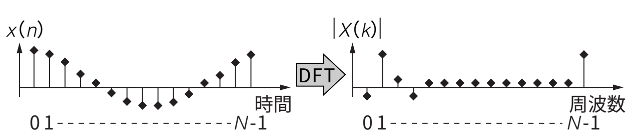


図2　離散フーリエ変換のイメージ

図の左が解析する信号．1周期をN点に分割している．右側が離散フーリエ変換の結果．信号の周波数成分を示している．

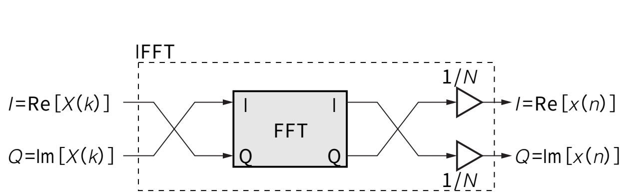


図3　逆高速フーリエ変換（IFFT）と高速フーリエ変換（FFT）

FFT回路があれば，IFFTを簡単に実現できる．

● **FFT演算の中身**

さて，今回設計するFFTの式について考えます．ここでは式を回転因子（twiddle factor）W_Nを用いて変形します．

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0,\ 1,\ \ldots,\ N-1)$$

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)} \qquad \cdots(2)$$

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot W_N^{nk} \quad (k = 0,\ 1,\ \ldots,\ N-1)$$

ここで2次元平面を用いて，回転因子W_Nを説明します．

オイラーの公式より，

$$e^{j\theta} = \cos\theta + j\sin\theta \qquad \cdots\cdots(3)$$

が成立します．この式を，X軸を実数（Re；real）軸，Y軸を虚数（Im；imaginary）軸とする2次元平面に示すと**図4**のようになります．すなわち，$e^{j\theta}$という関数はθを変化させることによって，半径1の円周を回転する点を示すことになります．フーリエ変換ではこの複素平面における回転を使って波形を解析します．したがって，反時計回りの回転は正の周波数に対応し，時計回りの回転は負の周波数に対応することになります．

回転因子W_Nを複素平面に示すと**図5**のようになります．FFTの式では，この回転因子W_Nは累乗になります．指数が増加すると，図に示すように，複素平面で時計方向に回転することになります．

FFT演算をさらに理解するために，4点FFTを行列表現したものを**図6**に示します．$N = 4$なので，4×4の回転因子が並ぶ行列ができます．回転因子は，**図6**の右側に示

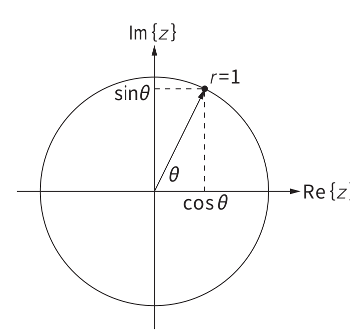


図4
オイラーの法則と2次元複素平面における表示

X軸は実数（Re；real）軸，Y軸は虚数（Im；imaginary）軸である．$e^{j\theta}$という関数はθを変化させることによって，半径1の円周を回転する点を示す．

$$W_N = e^{-j\left(\frac{2\pi}{N}\right)} = \cos\left(\frac{2\pi}{N}\right) - j\sin\left(\frac{2\pi}{N}\right)$$

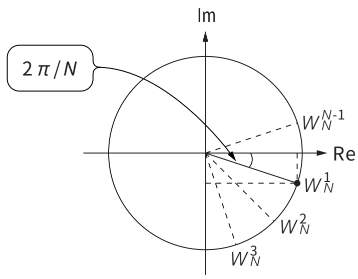


図5　回転因子 W_N の意味

指数が増加すると，複素平面で時計方向に回転する．

$$W = e^{-j\left(\frac{2\pi}{4}\right)}$$

$$X(k) = \sum_{n=0}^{3} x(n) \cdot W^{nk} \left(k = 0, 1, 2, 3\right)$$

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \end{bmatrix} = \begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix}$$

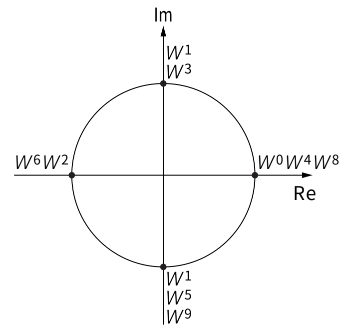


図6　4点FFTの行列表現

4×4の回転因子が並ぶ行列になる．W^0，W^1，…の順で複素平面状の半径1の円周を回転する．

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} =$$

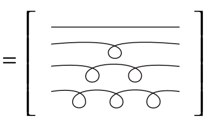


図7　4点FFTの行列に値を入れる

1行目は同じ値が続く周波数0の回転，2行目は複素平面状における負の方向への1回転，3行目は複素平面上における2回転，4行目は負の方向への3回転または正の方向への1回転と考えることができる．

すように，W^0，W^1，…の順で複素平面状の半径1の円周を回転します．

この回転因子の行列に数値を入れたものを**図7**に示します．1行目は1が並び，同じ値が続く周波数0の回転に対応しています．2行目は1，$-j$，-1，jとなり，複素平面上における負の方向へ(反時計回り)の1回転を示しています．3行目は1，-1，1，-1となり，複素平面上における2回転を示しています．4行目は1，j，-1，$-j$となり，負の方向への3回転であり，正の方向へ(時計回り)の1回転と考えることもできます．

すなわち，FFT演算とは複素数による回転を示す波形の周波数の異なるものそれぞれと，入力信号$x(n)$の内積計算をしていることになります．

●基底4における64点FFT演算アーキテクチャ

FFT演算の教科書では，バタフライ演算がよく使われています．とくにRADIX-2バタフライという演算が使用されています．しかし，ここでは現実的に実装の観点から効率の良いRADIX-4(基底4)バタフライ演算を前提に，64点FFT回路の演算方式を紹介します．

$64 = 4^3$なので，FFTのインデックスを以下のように変数変換します．各Σの加算個数は4個で，3重のΣからなる式に変形できます．

$$X(k) = \sum_{n=0}^{63} x(n) \cdot W_{64}^{nk}$$

$$k = k_0 + 4k_1 + 16k_2 \quad \left(k_0,\ k_1,\ k_2 = 0\sim3\right)$$

$$n = n_0 + 4n_1 + 16n_2 \quad \left(n_0,\ n_1,\ n_2 = 0\sim3\right)$$

$$X(k) = \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\sum_{n_2=0}^{3} x\left(n_0 + 4n_1 + 16n_2\right) W_{64}^{\left(n_0+4n_1+16n_2\right)\left(k_0+4k_1+16k_2\right)} \quad \cdots\cdots(4)$$

この一つのΣが1ステージの処理となり，3段で全体の処理が行えます．

1)ステージ1

最初に，n_2に関するΣを展開し，変形します．ここで，回転因子W_Nの性質，$(W_N)^0 = 1$，$(W_N)^{N/4} = -j$，$(W_N)^{2N/4} = -1$，$(W_N)^{3N/4} = j$を用います．

$$X(k) = \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\left\{ x\left(n_0 + 4n_1\right) W_{64}^{\left(n_0+4n_1\right)\left(k_0+4k_1+16k_2\right)} + x\left(n_0 + 4n_1 + 16\right) W_{64}^{\left(n_0+4n_1+16\right)\left(k_0+4k_1+16k_2\right)} \right.$$
$$x\left(n_0 + 4n_1 + 32\right) W_{64}^{\left(n_0+4n_1+32\right)\left(k_0+4k_1+16k_2\right)}$$
$$\left. + x\left(n_0 + 4n_1 + 48\right) W_{64}^{\left(n_0+4n_1+48\right)\left(k_0+4k_1+16k_2\right)} \right\}$$

$$= \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\left\{\left(x(n_0+4n_1) + (-j)^{k_0}x(n_0+4n_1+16) + (-1)^{k_0}x(n_0+4n_1+32) + (j)^{k_0}x(n_0+4n_1+48)\right)W_{64}^{k_0(n_0+4n_1)}\right\}W_{16}^{(n_0+4n_1)(k_1+4k_2)}$$

$$\cdots\cdots\cdots\cdots(5)$$

ここで，以下のように$x1$を考えます．

$$x1(n_0+4n_1+16k_0) = \left(x(n_0+4n_1) + (-j)^{k_0}x(n_0+4n_1+16) + (-1)^{k_0}x(n_0+4n_1+32) + (j)^{k_0}x(n_0+4n_1+48)\right)W_{64}^{k_0(n_0+4n_1)}$$

$$X(k) = \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}x1(n_0+4n_1+16k_0)W_{16}^{(n_0+4n_1)(k_1+4k_2)}$$

$$\cdots\cdots\cdots\cdots(6)$$

$x1$は四つの値に，jや-1などを掛けて加算した後に回転因子を乗算しています．この$x1$の計算が第1ステージの演算となります．次に，この$x1$を入力として，上記計算を行えば$X(k)$が求まりますが，これは16点FFTの式となります．ただし，$k_0=0$，1，2，3の場合があり，16点FFTを4回計算する必要があります．

2）ステージ2

第2ステージでは，ステージ1と同じ計算方法により，$x2$を求めます．

$$X(k) = \sum_{n_0=0}^{3}\sum_{n_1=0}^{3}x1(n_0+4n_1+16k_0)W_{16}^{(n_0+4n_1)(k_1+4k_2)}$$

$$= \sum_{n_0=0}^{3}\left\{x1(n_0+0+16k_0)W_{16}^{(n_0+0)(k_1+4k_2)} + x1(n_0+4+16k_0)W_{16}^{(n_0+4)(k_1+4k_2)} + x1(n_0+8+16k_0)W_{16}^{(n_0+8)(k_1+4k_2)} + x1(n_0+12+16k_0)W_{16}^{(n_0+12)(k_1+4k_2)}\right\}$$

$$= \sum_{n_0=0}^{3}\left\{\left(x1(n_0+0+16k_0) + (-j)^{k_1}x1(n_0+4+16k_0) + (-1)^{k_1}x1(n_0+8+16k_0) + (j)^{k_1}x1(n_0+12+16k_0)\right)W_{16}^{k_1n_0}\right\}W_4^{n_0k_2}$$

$$x2(n_0+4k_1+16k_0) = \left(x1(n_0+0+16k_0) + (-j)^{k_1}x1(n_0+4+16k_0) + (-1)^{k_1}x1(n_0+8+16k_0) + (j)^{k_1}x1(n_0+12+16k_0)\right)W_{16}^{k_1n_0}$$

COLUMN

▶▶▶ 第10回を迎えて

「LSIデザインコンテストin沖縄」も早いもので，今回で10年目を迎えることになりました．

1998年の第1回は，琉球大学工学部情報工学科内部の学生コンテストでした．その後，回を重ねるごとに学外，国外と参加者の枠が広がりました．ここ数年は，約100名の学生の参加があります．3月には10件前後の優秀デザインを集めた発表会を沖縄で開催し，おおいに盛り上がっています．2001年度より本誌との協力を開始し，学生・社会人がともに参加できるDesign Wave設計コンテストとしては第7回となります．

設計するテーマも時代を反映した内容を努めて出題するように心がけてきました．第1回のコンテストを立ち上げてくださったのは，当時，琉球大学工学部情報工学科学科長の翁長健治教授（現琉球大学名誉教授）とLSI関連教育担当の尾知博教授（現九州工業大学教授）でした．当時，筆者はまだ三菱電機の社員でした．1999年5月に琉球大学に転職してから，このコンテストを引き継ぎ運営させていただいています．

表A-1にこれまでの設計課題を示します．

表A-1 設計コンテスト10年間の課題

回	西　暦	設計テーマ	応　用
第1回	1998年	KUE-チップ	マイクロプロセッサ
第2回	1999年	RSA暗号デコーダ	暗号
第3回	2000年	リードソロモンCODEC演算エンジン	CD，デジタル通信／放送用エラー訂正
第4回	2001年	デジタルCDMAレシーバー	携帯電話、無線LAN
第5回	2002年	差集合巡回符号エラー訂正回路	TVの文字放送，地上デジタル放送
第6回	2003年	静的ハフマン符号用可変長デコーダ	音声圧縮，画像圧縮，動画圧縮
第7回	2004年	共通鍵暗号AES用SubByte変換回路	無線LAN，通信暗号化
第8回	2005年	デジタルFMレシーバー	無線通信・放送
第9回	2006年	2次元積符号繰り返しデコーダ	広域無線LAN
第10回	2007年	64点高速フーリエ変換回路	信号解析，OFDM通信，無線LAN

COLUMN

$$X(k)=\sum_{n_0=0}^{3} x2\left(n_0+4k_1+16k_0\right)W_4^{n_0k_2} \quad \cdots\cdots\cdots\cdots(7)$$

これをRADIX-4バタフライ回路と言います．

3）ステージ3

さらに式を変形します．

$$x3\left(k_2+4k_1+16k_0\right)=\sum_{n_0=0}^{3} x2\left(n_0+4k_1+16k_0\right)W_4^{n_0k_2}$$

$$=\left\{x2\left(0+4k_1+16k_0\right) +x2\left(1+4k_1+16k_0\right)W_4^{k_2} +x2\left(2+4k_1+16k_0\right)W_4^{2k_2} +x2\left(3+4k_1+16k_0\right)W_4^{3k_2}\right\}$$

$$=\left\{x2\left(0+4k_1+16k_0\right) +(-j)^{k_2}x2\left(1+4k_1+16k_0\right) +(-1)^{k_2}x2\left(2+4k_1+16k_0\right) +(j)^{k_2}x2\left(3+4k_1+16k_0\right)\right\} \quad \cdots\cdots\cdots\cdots(8)$$

前述のバタフライ演算と同じですが，回転因子の乗算が消えました．

4）リオーダ

$x3(k_2+4k_1+16k_0)$が求まりました．しかし本来のインデックス値は$(k_0+4k_1+16k_2)$です．インデックス値が入れ替わっています．

インデックス値の正しい出力を求めるためには，以下に示す関係を用いて，並べ替える必要があります．

$$X\left(k_0+4k_1+16k_2\right)=x3\left(k_2+4k_1+16k_0\right) \quad \cdots\cdots\cdots\cdots(9)$$

この計算のためのアーキテクチャを**図8**に示します．

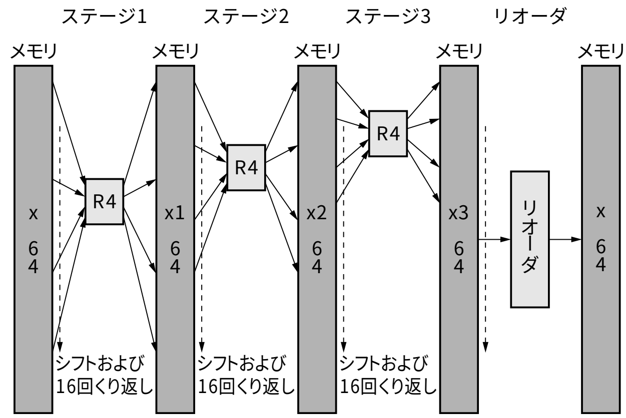


図8　RADIX-4の64点FFT演算アーキテクチャ
3段の演算の後，リオーダを行う．R4はRADIX-4のバタフライ演算．

● MATLABとScilabによるRADIX-4 64点FFT

FFT演算の理解のためには，MATLABのコードが役立ちます．RADIX-4による64点FFT演算を行うMATLAB関数ファイル（myfft64.m）を本コンテストのWebサイト（http://www.cqpub.co.jp/dwm/contest/）からダウンロードできます．

MATLABでは配列のインデックスが0ではなく1から始まります．x，$x1$，$x2$，$x3$，yの値は，インデックス1が最初になるので注意してください．

今回のコンテストでは，このMATLABコードを回路で実現することになります．FFTを完全に理解できない場合でも，myfft64.mを仕様として設計を進めることができます．

フリーの科学技術演算ソフトウェアScilabによるRADIX-4による64点FFTの処理を行う関数（myfft64.sce）も本コンテストのWebサイトからダウンロードできます．Scilabの詳細は，ScilabコンソーシアムのWebサイト（http://www.scilab.org/）を参照してください．

2．シリアル処理による64点FFT回路

64点FFTの一つの実装方法として，シフト・レジスタを用いるシリアル処理のFFTを説明します．

64点FFTの処理アルゴリズムから，三つのステージによる処理とリオーダが必要なことは理解されたと思います．

ステージ1では，16点ずつ離れた4点から16点ずつ離れた4点の出力を計算しています．ステージ2では4点ずつ離れた4点から4点ずつ離れた4点の出力を計算しています．ステージ3では隣接の4点から隣接の4点を計算します．このような処理は，シフト・レジスタを用いて処理することができます．

● ステージ3の4入力シリアルFFT

図9はステージ3における連続4点のFFTを示したものです．左側の正方形はフリップフロップです．合計7段のシフト・レジスタを構成しています．クロックごとに$x2$入力をシフトしていきます．中段の切り替え回路（マルチプレクサ）により，RADIX-4バタフライ演算回路の入力に四

つの$x2$入力信号が渡されます．RADIX-4バタフライ演算回路はこの入力に対してステージ3に必要な演算を行います．各入力に1，j，-1，$-j$を乗算して加算を行います．虚数単位jの乗算は，複素数の実数成分と虚数成分の交換や符号反転で実現できます．したがって，RADIX-4バタフライ演算回路には乗算は必要なく，加算器などで構成されます．最終出力に複素数の回転因子が複素乗算されています．ステージ3では実際には回転因子は1なので，乗算の必要はありません．

図10に**図9**の動作波形を示します．HEAD信号が‘1’で入力の開始を示します．IN信号の最初の4点が最初にFFTすべき入力値です．R0からR6はシフト・レジスタの値です．値がシフトしていくようすがわかります．PHASE信号はHEAD信号から生成され，S1，S2，S3のスイッチ信

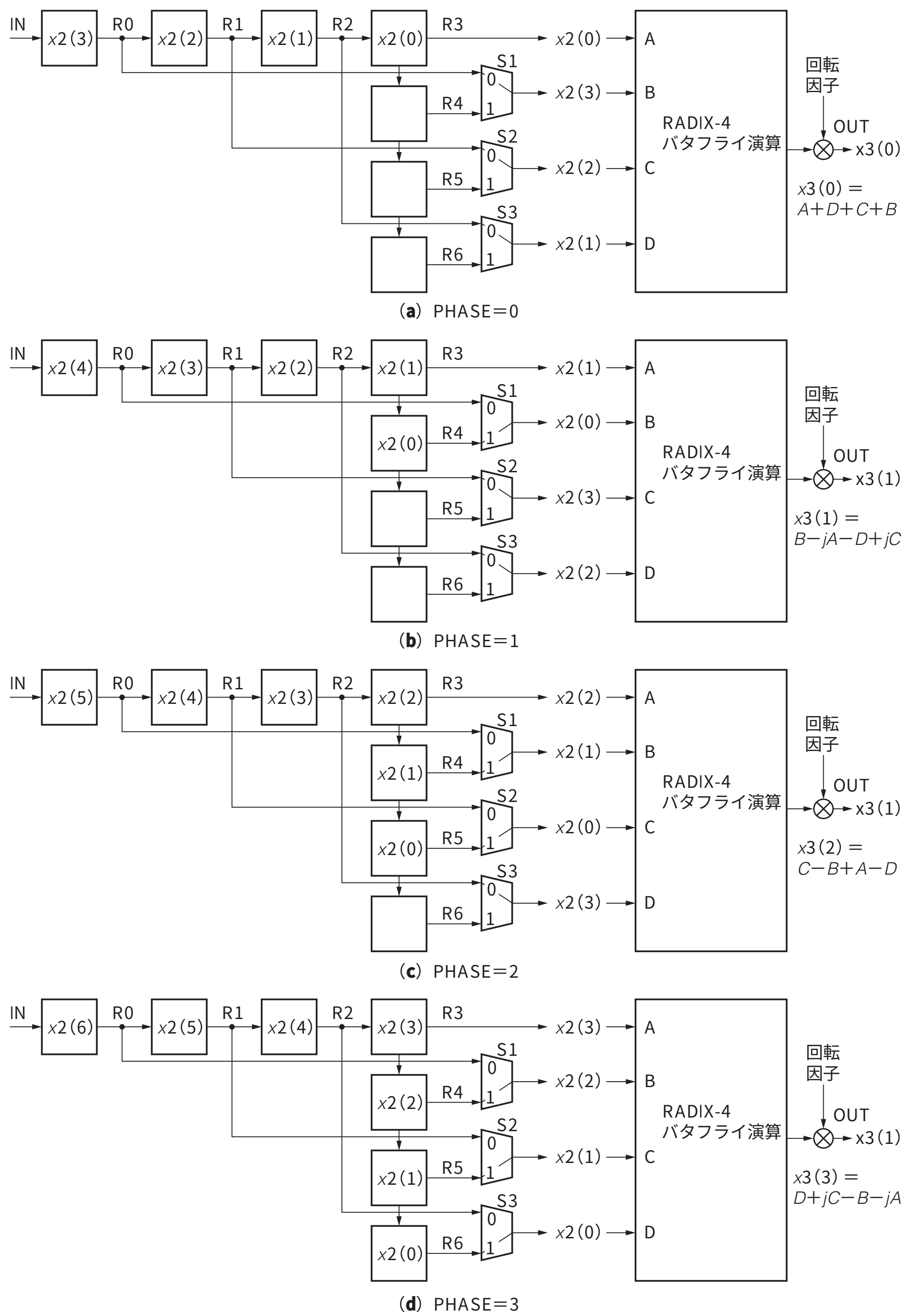


図9
連続4点のFFT
ステージ3における処理を示す．

号を生成します．結果的に，A，B，C，Dに4点の入力値が現れ，RADIX-4バタフライ演算回路からOUT信号が出力されます．また，OUT出力の先頭を示すために，OUTHEAD信号が示されています．OUTHEAD信号はHEAD信号をこの処理のレイテンシのサイクル遅延したものになっています．

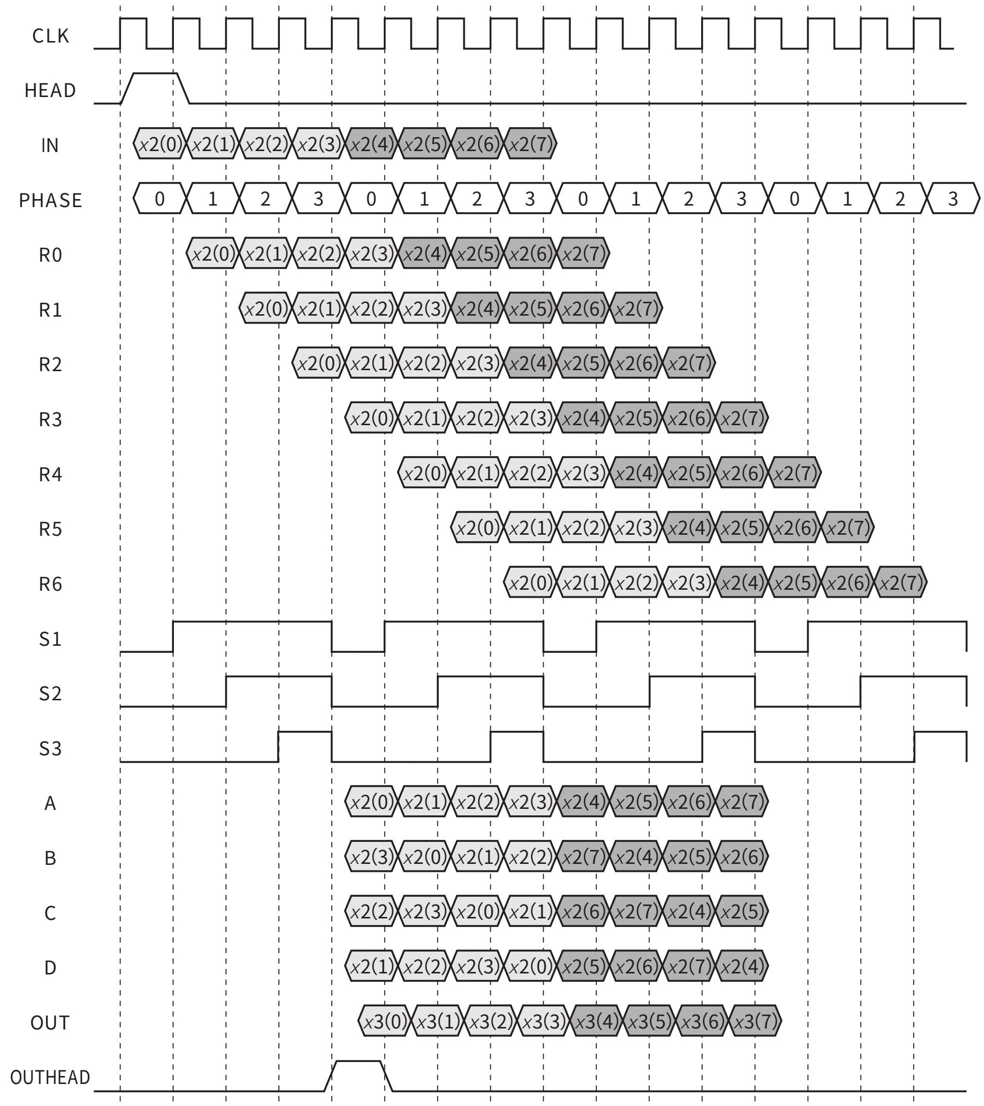


図10
連続4点のFFTの動作
IN信号の最初の4点が最初にFFTすべき入力値．R0からR6のシフト・レジスタとS1，S2，S3のスイッチ信号によりA，B，C，Dに4点の入力値が現れる．RADIX-4バタフライ演算回路からOUT信号が出力される．

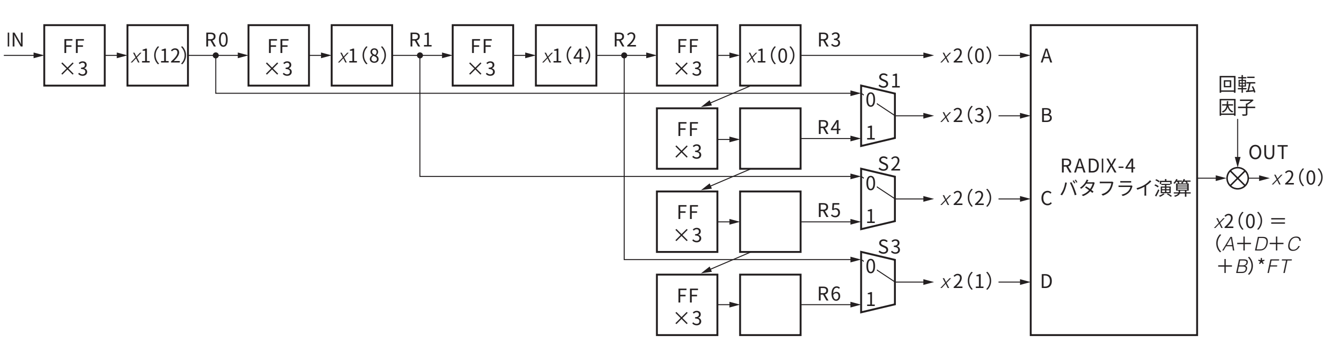


図11　ステージ2の処理
PHASE＝0を示す．ステージ3のシフト・レジスタの長さを変更し，やや制御を変更することで実現できる．

● ステージ1とステージ2の処理

ほかのステージの処理は，ステージ3のシフト・レジスタの長さを変更し，やや制御を変更することで実現できます．また，回転因子の複素数を生成して，複素乗算することも必要です．

参考までに，ステージ2の場合を**図11**に示します．これから容易にステージ1の処理が類推できます．

ステージ2の場合，4点おきの入力点を使用するので，シフト・レジスタのR0，R1，R2，R3，R4，R5，R6間の段数は4段となります．また，連続の4点を処理している間はPHASEが同じであり，4入力点ごとにPHASEをカウントアップします．あとはステージ3と同じですが，最終段にて回転因子を乗算します．

ステージ1では，シフト・レジスタのR0，R1，R2，R3，R4，R5，R6間の段数は16段となります．

● リオーダ処理

ステージ3の出力は，

$$X\left(k_0+4k_1+16k_2\right)=x3\left(k_2+4k_1+16k_0\right) \quad \cdots\cdots\cdots\cdots(10)$$

の$x3(0)$，$x3(1)$，$x3(2)$，$x3(4)$，…，の順に出力されます．これは本来のFFT出力に変換すると，$X(0)$，$X(16)$，$X(32)$，…，になります．**表1**に対応表を示します．

このリオーダには，フリップフロップなどのメモリを用いる必要があります．表をよく見ると規則性があるので，実際にはRAM（random access memory）やシフト・レジスタなどでも，リオーダ処理を実現できそうなことがわかります．

3．演算回路の実現法

● 固定小数点の表現

今回の課題では0.0625のような小数点以下の数をディジタル回路で取り扱う必要があります．例えば4ビットの2進数“0111”の場合，小数点をどこの位置に置くかによって表す値が変わってきます．“01.11”なら10進数表現では＋1.75，“0111.”なら10進数表現で＋7です．

また，4ビットの2進数では，それが符号なしの数で正または0を表しているか，2の補数表現で正または負の数を表しているかを区別する必要があります．“11.10”は符号なしであれば10進数表現で＋3.50となり，2の補数表現であれば10進数表現で－0.50となります．

このように同じ4ビットの2進数でも，「小数点の位置」と「符号なし表現か2の補数表現か」を明確にしないと，まったく異なった数になってしまいます．

ここでは，米国CoWare社のディジタル信号処理設計ツール「SPD（Signal Processing Designer；旧SPW：Signal Processing Workbench）」で用いられている固定小数点の属性表記方法を解説します．

＜8，2，t＞

と表記すると，その信号は，

8　：　全体のビット数が8ビット

表1　リオーダ

$x3(k_2+4k_1+16k_0)$	$X(k_0+4k_1+16k_2)$	$x3(k_2+4k_1+16k_0)$	$X(k_0+4k_1+16k_2)$	$x3(k_2+4k_1+16k_0)$	$X(k_0+4k_1+16k_2)$	$x3(k_2+4k_1+16k_0)$	$X(k_0+4k_1+16k_2)$
0	0	16	1	32	2	48	3
1	16	17	17	33	18	49	19
2	32	18	33	34	34	50	35
3	48	19	49	35	50	51	51
4	4	20	5	36	6	52	7
5	20	21	21	37	22	53	23
6	36	22	37	38	38	54	39
7	52	23	53	39	54	55	55
8	8	24	9	40	10	56	11
9	24	25	25	41	26	57	27
10	40	26	41	42	42	58	43
11	56	27	57	43	58	59	59
12	12	28	13	44	14	60	15
13	28	29	29	45	30	61	31
14	44	30	45	46	46	62	47
15	60	31	61	47	62	63	63

2　：　整数ビットが2ビット
t　：　2の補数表現（two's complement）
　　　すなわち最上位ビットは符号ビットとなる

という意味になります．したがって，“01101111”という8ビットの数の属性が＜8，2，t＞とすると，上位ビットから，“0”が符号ビット，“11”が整数部，“01111”が小数部となり，10進表現では＋3.46875に対応します．

＜8，2，u＞

と表記すると，その信号は，

8　：　全体のビット数が8ビット
2　：　整数ビットが2ビット
u　：　符号なし数（unsigned）
　　　すなわち負の数を表さない

という意味です．したがって，“01101111”という8ビットの数の属性が＜8，2，u＞とすると，上位ビットから，“01”が整数部，“101111”が小数部となり，10進表現では＋1.734375に対応します．

整数部のビット数が多いほど表現できる最大数，すなわち範囲が大きくなります．小数部のビット数が多いほど表現できる最小数が小さくなり，解像度が向上します．

4ビットの数が＜4，2，u＞の属性を持つ場合と＜4，1，t＞の属性を持つ場合の10進表現を**表2**に示します．また，4ビット長の固定小数点について，属性の例と対応する2進数，値の範囲，解像度を**表3**に示します．4ビット長であっても，多様な範囲と解像度の数値を表現できます．当然のことですが，解像度を悪く（大きな値に）すると，より広い範囲を表せます．

表2　属性による値の違い

4ビットの数	属性〈4,2,u〉の場合の10進表現	属性〈4,1, t〉の場合の10進表現
0000	＋0.00	＋0.00
0001	＋0.25	＋0.25
0010	＋0.50	＋0.50
0011	＋0.75	＋0.75
0100	＋1.00	＋1.00
0101	＋1.25	＋1.25
0110	＋1.50	＋1.50
0111	＋1.75	＋1.75
1000	＋2.00	－2.00
1001	＋2.25	－1.75
1010	＋2.50	－1.50
1011	＋2.75	－1.25
1100	＋3.00	－1.00
1101	＋3.25	－0.75
1110	＋3.50	－0.50
1111	＋3.75	－0.25

● 複素数の加算

二つの実数をa，bとすると，$a+jb$で複素数を示すことができます．jは，

$$j=\sqrt{-1} \qquad (11)$$

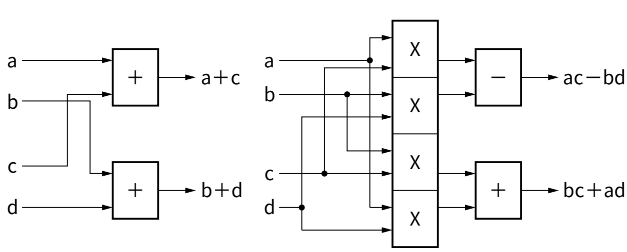
であり，虚数単位です．

二つの複素数$a+jb$と$c+jd$の加算は，実数成分と虚数成分のそれぞれの和になり，二つの実数加算器（通常の加算器）で実現できます．

$$(a+jb)+(c+jd)=(a+c)+j(b+d) \qquad (12)$$

表3　4ビット固定小数点の属性と対応する2進数，値の範囲，解像度

	属　性	2進数表現（Sは符号ビット　Xはデータ・ビット）	範　囲	解像度
整　数	〈1,1,u〉	X.	0 ～ 1	1
	〈4,4,u〉	XXXX.	0 ～ 15	1
	〈4,3,t〉	SXXX.	－8 ～ 7	1
小　数	〈4,0,u〉	.XXXX	0.0 ～ 0.9375	0.0625（1/16）
	〈4,0,t〉	S.XXX	－1.00 ～ +0.875	0.125（1/8）
その他	〈4,2,u〉	XX.XX	0.0 ～ 3.75	0.25（1/4）
	〈4,2,t〉	SXX.X	－4.0 ～ +3.5	0.5（1/2）
	〈4,5,u〉	XXXX0.	0 ～ 30	2
	〈4,5,t〉	SXXX00.	－32 ～ 28	4
	〈4,－1,u〉	.0XXXX	0.0 ～ 0.46875	0.03125（1/32）
	〈4,－1,t〉	S.SXXX	－0.5 ～ +0.4375	0.0625（1/16）



（**a**）加算　　（**b**）乗算

図12　複素数演算回路

複素加算器は2セットの二つの実数が入力され，それぞれが加算されて，二つの実数を出力する組み合わせ回路．複素乗算器も同じ．

● 複素数の乗算

二つの複素数$a+jb$と$c+jd$の乗算は，

$$(a+jb)\times(c+jd)=(ac-bd)+j(bc+ad)\ \cdots\cdots\cdots\cdots(13)$$

のようになります．したがって，複素乗算は四つの実数乗算と一つの減算，一つの加算で実現できます．

複素数演算の回路構成を**図12**に示します．複素数$a+jb$であっても回路の中では二つの信号a，bが存在するだけです．複素加算器は2セットの二つの実数が入力され，それぞれが加算されて，二つの実数を出力する組み合わせ回路です．複素乗算器も同様です．

● 回転因子ROM

今回は**図13**に示すW_{64}の0乗から63乗までの値が回転因子として必要です．実数成分はcos関数で，虚数成分は(－sin)関数ですが，それぞれ64種類なので，アドレスとして6ビットの入力を持つ2個のROMで実現できます．

$$W_{64}=e^{-j\left(\frac{2\pi}{64}\right)}$$
$$=\cos\left(\frac{2\pi}{64}\right)+j\left[-\sin\left(\frac{2\pi}{64}\right)\right]$$

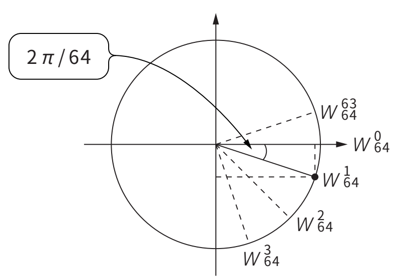


図13　回転因子
W_{64}の0乗から63乗までの値が必要．

64ワード×8ビットのROMのデータは，本コンテストのWebサイトからダウンロードできます．0～63のアドレス（address）に対応して，1周期分のcosと－sinの値，すなわち，

real ＝ cos（2×π×address/64）
imag ＝ －sin（2×π×address/64）

の10ビットの値が属性＜10，0，t＞で格納されています．三角関数の値なので，プログラムなどで簡単に生成することもできます．

4．課題

● LEVEL1：基本課題

基本課題では，**表4**に示すピン配置に合わせて回路を設計してください．シリアル入力のFFTIN信号とHEAD信

表4　基本課題用ピン配置

FFT			
信号名	入出力	ビット幅	説　明
CLK	IN	1	クロック入力
HEAD	IN	1	‘1’で先頭を示す
FFTIN_I	IN	8	入力の実数成分，〈8,0,t〉フォーマットとする
FFTIN_Q	IN	8	入力の虚数成分，〈8,0,t〉フォーマットとする
OUTHEAD	OUT	1	‘1’で先頭を示す
FFTOUT_I	OUT	14	出力の実数成分，〈14,6,t〉フォーマットとする
FFTOUT_Q	OUT	14	出力の虚数成分，〈14,6,t〉フォーマットとする

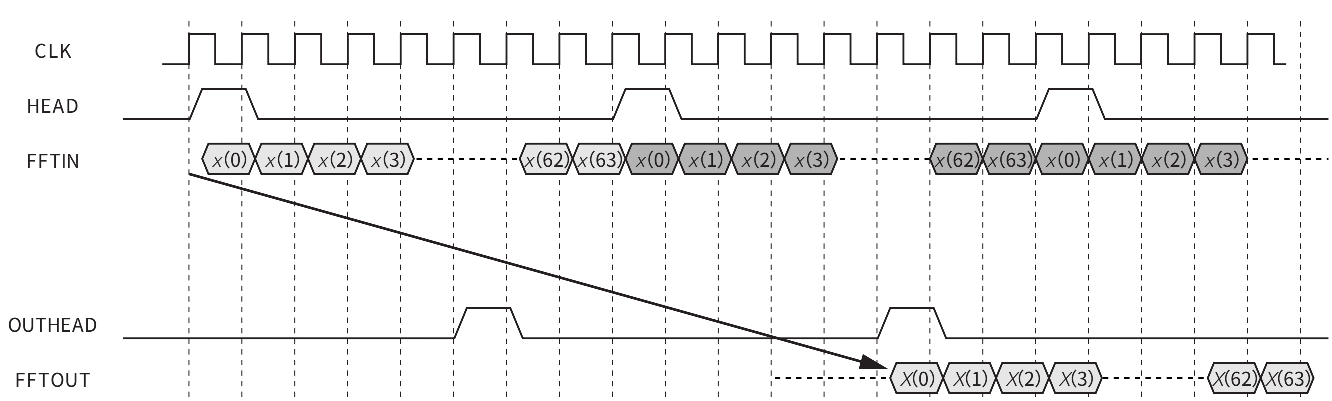


図14　基本課題の信号のタイミング
64点が休みなく入力され，その先頭位置はHEAD信号で示される．FFTのレイテンシ（遅延）は任意．

号をもとにFFTを行い，FFTOUT信号とその先頭位置を示すOUTHEAD信号を出力します．

タイミングを**図14**に示します．64点が休みなく入力され，その先頭位置はHEAD信号で示されます．図中の太い矢印はFFTのレイテンシ（遅延）を示していますが，レイテンシは任意です．

以下では，5種類の入力波形とそのFFT結果の例を紹介します．

1）すべて1を入力

図15（a）は，入力として1＋0*j*を64点入力した例です．そのFFT出力は2番目の波形です．インデックス0番におおよそ64.0の値が出力され，インデックス2以降は0です．

2）1周期の複素回転信号　初期値＝1＋0*j*

図15（b）は，ちょうど1周期の複素回転波形を入力した例です．FFT出力はインデックス1におおよそ64.0の値が出力されています．回転の初期値が1＋0*j*ですので，FFT出力はこの64倍の値になっています．

3）1周期の複素回転信号　初期値＝0－1*j*

図15（c）は，ちょうど1周期の複素回転波形を入力した例です．ただし，**2）**とは初期値が異なっています．FFT出力はインデックス1におおよそ－64.0の値が出力されています．回転の初期値が0－1*j*なので，FFT出力はこの64倍の値になっています．

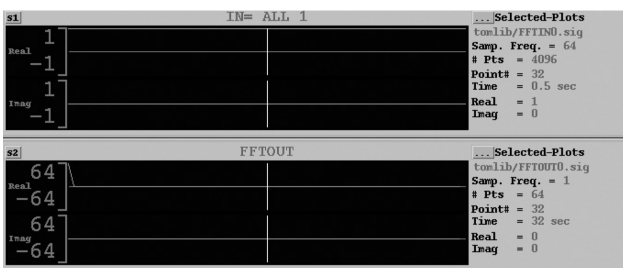
4）1周期のcos信号

図15（d）は，実数のcos信号です．虚数成分は0です．

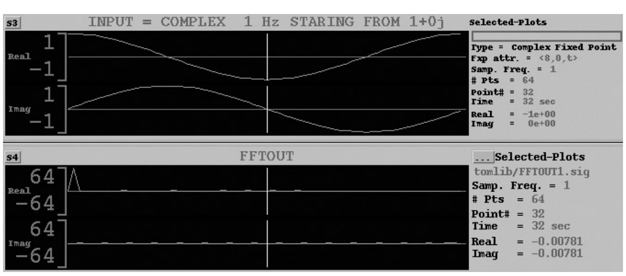
$$\cos\ \theta = \frac{e^{j\theta}+e^{-j\theta}}{2} \cdots\cdots (14)$$

1Hzのcos関数は，1Hzと－1Hzの成分を持ち，それぞれの大きさが0.5である二つの複素の回転の合成と考えることができます．したがって，FFT出力はインデックス1とインデックス63に，これまでの半分の大きさである32.0の値が出力されています．

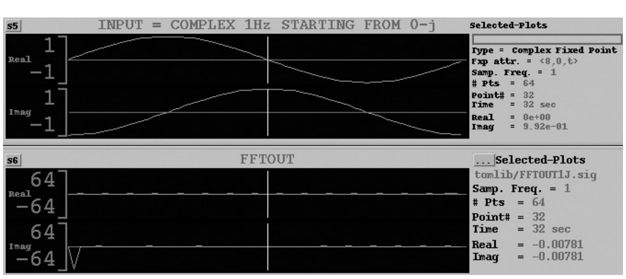
FFTでは周期（＝64）性より，インデックス63は－1Hz



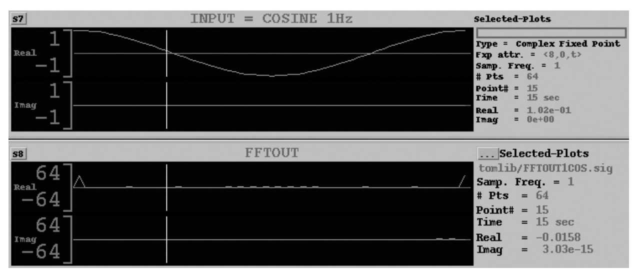
（**a**）すべて1を入力



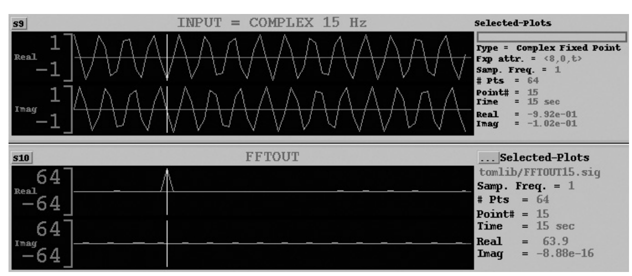
（**b**）1周期の複素回転信号　初期値＝1＋0*j*



（**c**）1周期の複素回転信号　初期値＝0－1*j*



（**d**）1周期のcos信号



（**e**）15周期の複素回転信号

図15　入力波形とそのFFT結果

に対応すると考えることができます．

5）15周期の複素回転信号

図15（e）は，ちょうど15周期の複素回転波形を入力した例です．FFT出力はインデックス15に，おおよそ64.0の値が出力されています．回転の初期値が1＋0*j*なので，FFT出力はこの64倍の値になっています．

五つの入力波形の実数成分と虚数成分の値を＜8，0，t＞フォーマットにしたものを本コンテストのWebサイトからダウンロードできます．テスト動作のときに活用してください．

● LEVEL2：自由課題

基本課題では，クロックごとに入力信号が与えられていますが，自由課題では**図14**の波形などを自由に変更してください．

案としては，

1）クロック周波数を上げて，数クロックごとに1データの入力を行う．クロック周波数の高さを利用して，回路を削減する．

2）例では，シフト・レジスタを用いた設計例を示していますが，RAMを用いて設計する．

などです．64点FFTを実行するユニークな回路を設計してください．

リスト1　50入力XOR回路のVHDLソース・コード

```
library  IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.NUMERIC_STD.all;

entity PARITY is
    port ( A : in  unsigned(49 downto 0);
           Y : out std_logic );
end PARITY;

architecture RTL of PARITY is
begin

    process(A)
    variable TMP : std_logic;
    begin
        TMP := '0';
        for i in 0 to 49 loop
            TMP := TMP xor A(i);
        end loop;

      Y <= TMP;
    end process;
end RTL;
```

● 社会人部門の条件

社会人部門は，課題をより実践的にするため，FPGAに実装することを前提として設計していただきます．

レポートには，FPGA設計ツールが出力したレポート・ファイル（使用論理ブロック数や最大動作周波数がわかるもの）を添付してください．ターゲットFPGAの選択についても審査対象とします．設計意図，アーキテクチャ，コスト・パフォーマンスなどを考慮して，ターゲットFPGAを選択してください．

VHDL，Verilog HDL以外の設計言語による参加も認めます．

なお，2006年12月10日までの間に，FPGA評価ボードやFPGA開発ツールの貸し出しや応募時のレポート方法に関する情報の提供などを行う場合があります．本コンテストのWebサイトや本誌2006年12月号～2007年2月号に掲載する「コンテスト参加要領」をかならずご確認ください．

5．速度と規模の単位（学生部門対象）

筆者の勤務する大学では，論理合成用のツールとして米国Synopsys社の「Design Compiler」を使用していますが，このツールはだれもが使えるわけではありません．そこで，

- 2入力XORゲート一つの遅延時間を1UNIT_DELAY
- 2入力XORゲート一つの面積を1UNIT_AREA

とします．

具体的には，50入力XOR回路を合成していただき，その1段当たりの遅延時間を「単位時間」として，速度の単位とします．50入力XOR回路のVHDLソース・コードを**リスト1**に示します（このファイルも本コンテストのWebサイトからダウンロード可能）．

● 速度

筆者の使用しているライブラリを用いた例では，6段，49個のXOR回路が合成されます．クリティカル・パス遅延はreport_timingコマンドにより7.17であることがわかります．そこで7.17/6＝1.195を単位（1UNIT_DELAY）とします．例えば，ある遅延が20ならば，20/1.195＝17.74UNIT_DELAYということになります．

● 規模

筆者の使用しているライブラリを用いた例では，面積は

report_areaコマンドのtotal cell areaにより147.0であることがわかります．XORゲート数は49個なので，147.0/49＝3.0を単位（1UNIT_AREA）とします．例えば，ある回路面積が200ならば，200/3.0＝66.67UNIT_AREAとします．

● RAMやROMを用いる場合の注意

FFTでは，やや多めのデータを取り扱うので，メモリを使用する場合が想定されます．メモリを使う場合も，回路規模に含まれるように合成させてください．具体的には，メモリは合成可能なHDLで記述し，フリップフロップなどを用いる回路で実現してください．三角関数出力などでROMテーブルを使用する場合も，マクロなどは使用せず，組み合わせ回路で実現してください．

ただし，FPGAのみをターゲットとする場合は，内蔵のメモリ・ブロックを使用してもかまいません．その場合は，レポートにわかりやすく明示してください．

6．提出レポートについて

レポートには以下の内容を含めてください．また，ページ数は少なめに，コンパクトにまとめてください．

<表紙>

1）代表者の氏名，チーム名，社会人/大学院修士/大学学部生/高専生の区別
2）共同設計者全員の名まえ（最大3名まで），会社/学校名（学籍番号，学年），住所，電話番号，E-mailなどの連絡先
3）学生部門参加者のみ全員のTシャツの希望サイズ（S，M，L，XL）
4）取り組んだ課題（レベル1/レベル2）

<内容>

1）設計した回路ブロックの構成説明（ブロック図と説明）
2）設計した回路ブロックの動作説明（動作波形図やパイプライン動作などの説明）
3）くふうした点，オリジナリティを出した点（アピールが重要！）
4）クリティカル・パスの速度，論理合成後の回路規模
5）VHDLもしくはVerilog HDLのコード
6）正常動作しているVHDL/Verilog HDLシミュレーションの波形
7）そのほか自由意見など

レポートはPDFファイルを推奨します．PDFファイルを作成できない場合はご相談ください．

社会人か学生かによって評価の方法が異なります．それにともない，レポートの送付先が異なります．

社会人部門：contest.dwm@cqpub.co.jp
学生部門　：wada@ie.u-ryukyu.ac.jp

締め切りは2007年1月26日（金）必着です．

7．審査のポイント

速度，回路規模だけでなく，アーキテクチャのユニークさ，アイデア，おもしろさを十分考慮して審査します（ちゃんとアピールしてくださいね！）．

社会人は，Design Wave設計コンテスト審査委員会が審査を行います．学生は，琉球大学で1次審査を行い，最終審査に残ったチームは，2007年3月16日に沖縄産業支援センターで開催予定の発表会に招待され，その会場で最終審査を行います．大学院修士，学部生，高専生のレベルに応じて審査します．

仕様書に従ってまじめに作るのもけっこうですが，おもしろいアイデアを歓迎します．他人と違ったことをしよう！ 仕様の部分変更など，柔軟に受け付けます．

ENJOY HDL!　沖縄で会おう！

＊　＊　＊

本設計コンテストの学生部門は，主催：LSIコンテスト実行委員会，共催：琉球大学工学部 情報工学科，沖縄産業振興センター，九州半導体イノベーション協議会，協賛：ソニーLSIデザインにより実施されています．

わだ・ともひさ
琉球大学工学部情報工学科